

# An Efficient Partial Sums Generator for Constituent Code based Successive Cancellation Decoding of Polar Codes

Tiben Che and Gwan Choi

Department of Electrical and Computer Engineering  
Texas A&M University, College Station, Texas 77840  
Email: {ctb47321, gchoi}@tamu.edu

**Abstract**—This paper proposes an architecture of partial sum generator for constituent codes based polar code decoder. Constituent codes based polar code decoder has the advantage of low latency. However, no purposefully designed partial sum generator design exists that can yield desired timing for the decoder. We first derive the mathematical presentation with the partial sums set  $\beta^c$  which is corresponding to each constituent codes. From this, we concoct a shift-register based partial sum generator. Next, the overall architecture and design details are proposed. The overhead compared with conventional partial sum generator is evaluated as well. Finally, the implementation results with both ASIC and FPGA technology and relevant discussions are presented.

## I. INTRODUCTION

Recently, polar codes [1] have received increasing attentions due to that it is the first code which provably achieves the channel capacity. Its low-complexity encoding and decoding schemes makes it very promising for real application as well. Now mainly there are three kinds of algorithm for polar codes decoding. E. Arikan in [1] presents a successive cancellation (SC) algorithm which is able to successively accomplish decoding with recursive cancellation. I. Tal [2] makes the SC algorithm more competitive by exploring more paths among the codewords tree; and this method is referred as list successive cancellation (LSC). Also, N. Hussami et al. in [3] shows that the belief propagation (BP) can be applied as decoding algorithm.

Although many efforts have been done for BP decoder [4], [5] and [6], it still suffers from the problem of high computing complexity. Thus, SC and LSC attract more and more studies especially on their hardware architecture. C. Leroux [7] proposed a semi-parallel architecture for both tree and line SC decoder, which gives a good inspiration about the trade-off between hardware complexity and latency. C. Zhang [8] proposed a low latency SC decoder with pre-computing and overlapped architecture. B. Yuan [9] proposed an architecture which is able to further reduce the latency by applying the 2-bit decoding at the last stage and being with some gate level optimizations. B. Yuan [10] also proposed an architecture of LSC using multi-bit decision. T. Che [11] proposed an overlapped LSC design approach which can increase the hardware efficiency. Balatsoukas-Stimming [12] [13] provided the hardware architecture of LSC and LLR based LSC algorithm as well. LSC is an extension of the SC. They two share the same SC processing part. The improvement of SC decoder can also benefit LSC decoding. Thus, latency reduction on SC is crucial for both of them. Alamdar-Yazdi [14]

proposed the simplified SC (SSC) which can reduce the latency by finding some certain pattern sub-codewords. These kinds of sub-codewords are also called constituent codes. Sarkis [15] proposed the fast-SSC which can further reduce the latency by exploring more kinds of constituent codes. Based on their idea, T. Che [16] proposed the throughput centric SC (TCSC) architecture which can significantly reduce the latency with only a little bit hardware penalty.

SC decoding is based on the feedback from decoded codewords which is also called partial sum. A partial sum generator (PSG) is needed for each SC decoder. The partial sum needs to be calculated at the same clock cycle when the codewords is determined. Thus, it's on the critical path of the decoding and can affect the maximum frequency of the decoder. Some works have been done for a good PSG design. C. Leroux [7] proposed an indicator function based PSG (IF-PSG). C. Zhang [8] proposed a PSG with feedback part (FB-PSG). J. Lin [17] proposed a hybrid PSG for LSC. G. Berhault proposed a shift-register-based PSG (SR-PSG) [18] [19], which is able to increasing the timing performance and reduce the hardware complexity. Y. Fan [20] proposed a similar architecture with SR-PSG however with higher level simplification.

All the aforementioned PSGs are capable of increasing the timing performance of SC decoder. However, none of them has considered the constituent codes situation. Since introducing the concept of constituent codes into decoding processing can significantly reduce the latency, it is reasonable and necessary to design a constituent-codes-compatible PSG. In this paper, we proposed an efficient PSG for constituent code based SC decoding. Firstly, we derive the mathematical presentation for constituent based PSG. This derivation is based on the SR-PSG for conventional SC decoder. Next, the overall hardware architecture and design details are proposed. Finally, the implementation result are presented. This architecture is implemented with both VLSI and FPGA technology. The relevant discussions are also mentioned as well.

This paper is organized as follows. The relative background is reviewed in section II. In following, the proposed design including the mathematical derivation are described in section III. After that, the implement results and relevant discussions are presented in section IV. Finally, this paper is concluded in section V.

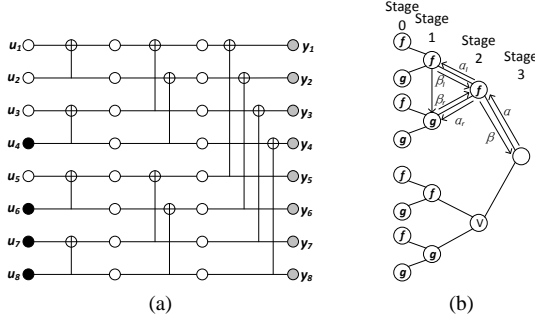


Fig. 1. (a) Encoder of (8, 4) polar code, (b) Tree presentation of (8, 4) SC decoder

## II. BACKGROUND

### A. Polar Code

As introduced by E. Arikan [1], we can construct polar code by successively performing channel polarization. Fig. 1a shows an example of the construction of 8-bit polar code. Mathematically, polar codes are linear block codes of length  $N = 2^n$ . The coded codeword  $\mathbf{x} \triangleq (x_1, x_2, \dots, x_N)$  is computed by  $\mathbf{x} = \mathbf{u}\mathbf{G}$  where  $\mathbf{G} = \mathbf{F}^{\otimes m}$ , and  $\mathbf{F}^{\otimes m}$  is the  $m$ -th Kronecker power of  $\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ . Each row of  $\mathbf{G}$  is corresponding to an equivalent polarizing channel. For an  $(N, k)$  polar code,  $k$  bits that carry source information in  $\mathbf{u}$  are called information bits. They are transmitted via the most  $k$  reliable channels. While the rest  $N - k$  bits, called frozen bits, are set to zeros and are placed at the least  $N - k$  reliable channels.

Polar codes can be decoded by recursively applying successive cancellation to estimate  $\hat{u}_i$  using the channel output  $y_0^{N-1}$  and the previously estimated bits  $\hat{u}_0^{i-1}$ . This method is referred as successive cancellation (SC) decoding. Actually, SC decoding can be regarded as a binary tree traversal as described in Fig. 1b. The number of bits of one node in stage  $m$  ( $m = 0, 1, 2, \dots$ )  $N^m$  is equal to  $2^m$ .  $\alpha$  stands for the soft reliability value, typically is log-likelihood ratio (LLR). Each left and right child nodes can calculate the LLR for current node via  $f$  and  $g$  functions, respectively [7]. However, in order to solve  $g$  function, a feedback  $\beta_l$  from left child of the same parent node is needed. This kind of feedback is called partial sum. At stage 0,  $\beta$  of a frozen node is always zero, and for information bit its value is calculated by threshold detection of the soft reliability according to

$$\beta = h(\alpha) = \begin{cases} 0, & \text{if } \alpha \geq 0 \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

At intermediate stages,  $\beta$  can be recursively calculated by

$$\beta[i] = \begin{cases} \beta_l[i] \oplus \beta_r[i] & \text{if } i \leq N^m/2 \\ \beta_r[i - N^m/2] & \text{otherwise} \end{cases} \quad (2)$$

### B. Constituent codes based SC decoding

SC decoding generally suffers from the high latency due to its inherent serial property. The processing of getting the partial sum from each node significantly constrain the decoding speed. Thus, in order to reduce the latency for calculating partial sum,

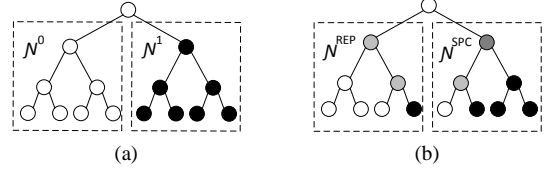


Fig. 2. SC decoding tree simplified by constituent codes

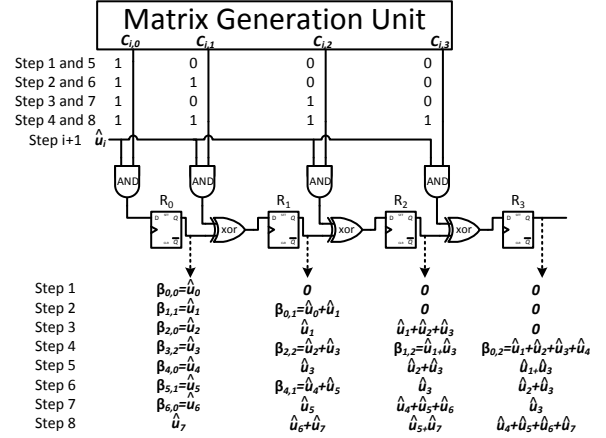


Fig. 3. The architecture of SR-PSG

constituent based SC decoding has been proposed [14], [15]. By finding some certain patterns in the source code, some part of the codeword and their corresponding partial sums can be estimated immediately without traversal. This method significantly reduces the partial-sum-constrained latency.  $\mathcal{N}^0$ ,  $\mathcal{N}^1$ ,  $\mathcal{N}^{SPC}$  and  $\mathcal{N}^{REP}$  are the four commonest constituent code.

$\mathcal{N}^0$  and  $\mathcal{N}^1$  only contain either frozen bits or information bits, respectively. For  $\mathcal{N}^0$  codes, we can set the corresponding to 0 immediately. For  $\mathcal{N}^1$  node, the partial sums can be directly determined via threshold detection Eq. (1).  $\mathcal{N}^{SPC}$  and  $\mathcal{N}^{REP}$  contain both frozen bits and information bits. In the  $\mathcal{N}^{SPC}$  codes, only the first bit is frozen. It makes the length  $N$  constituent codes as a rate  $(N - 1)/N$  single parity check (SPC) code. This code can be decoded by performing parity check with the least reliable bit. Typically it's the one with minimum absolute value of LLR. In the  $\mathcal{N}^{REP}$  codes, only the last bit is information bit. In this case, all the corresponding partial sums should be the same since they all are the reflection of the last information bit. Thus, the decoding algorithm starts by summing all input LLRs and the partial sums is calculated by performing the hard detection to the final summary. Fig. 2 shows an example of how constituent code can simplify the SC decoding tree. According to T. Che's implementation of constituent code based SC decoder [16], the latency of length  $N$  constituent code can be reduced from  $2N - 2$  to 1, 1,  $\log N + 1$  and  $\log N$  for  $\mathcal{N}^0$ ,  $\mathcal{N}^1$ ,  $\mathcal{N}^{SPC}$  and  $\mathcal{N}^{REP}$  codes, respectively. In order to further optimize the performance constituent codes based decoder, a specific designed PSG for it is very necessary.

### C. Shift-register-based partial sums generator

Among all the aforementioned PSG design, shift-register-based PSG (SR-PSG) has a better performance in terms of

the timing and hardware complexity. For length  $N$  polar code decoder, it consist of  $N$  registers and some other simple combination logic. Along with the estimation of each  $\hat{u}_i$ , the registers perform shift calculation and the partial sums can be obtains from their corresponding register. Its architecture is illustrated in Fig. 3. This architecture is built according to the following rule:

$$\begin{cases} R_0 \Leftarrow \hat{u}_i \cdot c_{i,0} \\ R_k \Leftarrow R_{k-1} \oplus (\hat{u}_i \cdot c_{i,k}), \text{ if } k \geq 0 \end{cases} \quad (3)$$

where  $\cdot$  and  $\oplus$  stand for *and* and *exclusive-or* operation, respectively. In Fig. 3,  $\hat{u}_i$  means the  $i$ th estimated bit.  $\beta_{i,j}$  means the  $j$ th partial sum in stage  $i$ .  $C_{i,j}$  means the  $i$ th row and  $j$ th column in the generate matrix  $G$ . The matrix generation unit is able to generate  $C_{i,j}$  with very simple logic. The SC decoder consists of many basic computation parts called processing unit (PU). Each partial sum needs to be feed into corresponding PU. The shift register based architecture can guarantee that all partial sum required by a PU are all generated in the same register, which can avoid any extra routing logic in the circuit.

Such architecture is able to receive the estimated bit and update the corresponding partial sum by every valid cycle, which keeps highly consistent with SC decoding processing. However, this architecture is not suitable for constituent codes based SC decoder since some partial sums are obtained directly instead of calculating from estimated bits. Thus, a PSG for constituent codes based SC decoder should have the capability to generate the new partial sums from either the directly got intermediate partial sums or the estimated bits, and to maintain the coherence of them.

### III. PROPOSED DESIGN

In this section, we first derive the mathematical presentation of constituent code based partial sum from Eq. (3). Then, the overall hardware architecture and relevant design details are presented.

#### A. Mathematical Presentation

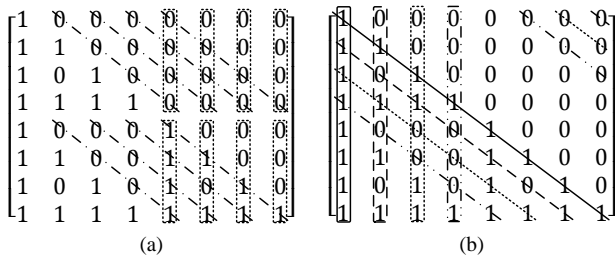


Fig. 4. (a) Elements shift in generation matrix, and (b) diagonal cycle-shift in generation matrix

For a length  $n$  constituent code, its corresponding estimated bits and partial sums are denoted as  $\hat{u}_{i-n+1}^c \dots \hat{u}_i^c$  and  $\beta_0^c \dots \beta_{n-1}^c$ , respectively. All the  $\beta^c$  are obtained at the same time. For those bits who are not belongs to any constituent codes, we still have to calculate their corresponding partial sums according to Eq. (3). Thus, if we still want to keep consistent between directly got intermediate partial sums and the

one-by-one-estimated bits, we need to derive the mathematical presentation with  $\beta^c$  from Eq. (3).

For  $k \geq n$  and  $k \in [a \cdot n, (a+1) \cdot n - 1]$ ,  $a = 1, 2, \dots$ , according to Eq. (3), we have

$$\begin{aligned} R_k &= R_{k-1} \oplus (\hat{u}_i^c \cdot c_{i,k}) \\ &= R_{k-2} \oplus (\hat{u}_{i-1}^c \cdot c_{i-1,k-1}) \oplus (\hat{u}_i^c \cdot c_{i,k}) \\ &\dots \\ &= R_{k-n} \oplus \\ &\quad \left( \begin{bmatrix} \hat{u}_{i-n+1}^c & \dots & \hat{u}_i^c \end{bmatrix} \begin{bmatrix} c_{i-n+1,k-n+1} \\ \vdots \\ c_{i,k} \end{bmatrix} \right). \end{aligned} \quad (4)$$

As we know,  $c_{i,k}$  is the element of generation matrix  $G$  which is the Kronecker power of  $F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ . Combine this property with our observation on the matrix, we conduct the following rule which is also noted in Fig. 4a.

$$\begin{bmatrix} c_{i-n+1,k-n+1} \\ \vdots \\ c_{i,k} \end{bmatrix} = \begin{bmatrix} c_{i-n+1,(a+1) \cdot n - (k \bmod n) - 1} \\ \vdots \\ c_{i,(a+1) \cdot n - (k \bmod n) - 1} \end{bmatrix}. \quad (5)$$

According to the definition of generation matrix and concept of constituent code, when  $c_{i,k} = 0$ , the right part of Eq. (5) is equal to a all **zero** vector, and when  $c_{i,k} = 1$  the right part of Eq. (5) is equal to the  $(n - (k \bmod n) - 1)$ th column in the generation matrix for length  $n$  polar code. According to the definition of partial sum and Eq. (2), we get

$$[\hat{u}_{i-n+1}^c, \dots, \hat{u}_i^c] \cdot [c_{i-n+1,p(k)}, \dots, c_{i,p(k)}]^T = \beta_{p(k)} \quad (6)$$

where  $p(k) = (n - (k \bmod n) - 1)$ .

Now we apply above observation back to Eq (4). We define the vector  $\mathbf{R}_a = [R_{a \cdot n}, \dots, R_{a \cdot n + n - 1}]$  and  $\mathbf{c}_{i,a} = [c_{i,a \cdot n}, \dots, c_{i,a \cdot n + n - 1}]$  for  $k \in [a \cdot n, (a+1) \cdot n - 1]$ ,  $a = 1, 2, \dots$ . We also define the vectors  $\hat{\mathbf{u}}^c = [\hat{u}_{i-n+1}^c, \dots, \hat{u}_i^c]$  and  $\hat{\beta}^c = [\beta_{n-1}^c, \dots, \beta_0^c]$ . Then, we have

$$\begin{aligned} \mathbf{R}_a &= [R_{a \cdot n}, \dots, R_{a \cdot n + n - 1}] \\ &= [R_{(a-1) \cdot n}, \dots, R_{a \cdot n - 1}] \oplus \\ &\quad \left( [\hat{u}_{i-n+1}^c, \dots, \hat{u}_i^c] \begin{bmatrix} c_{i-n+1,a \cdot n - n + 1} & \dots & c_{i-n+1,a \cdot n} \\ \vdots & \ddots & \vdots \\ c_{i,a \cdot n} & \dots & c_{i,a \cdot n + n - 1} \end{bmatrix} \right) \\ &= [R_{(a-1) \cdot n}, \dots, R_{a \cdot n - 1}] \oplus \\ &\quad \left( \hat{\mathbf{u}}^c \begin{bmatrix} c_{i-n+1,p(a \cdot n)} & \dots & c_{i-n+1,p(a \cdot n + n - 1)} \\ \vdots & \ddots & \vdots \\ c_{i,p(a \cdot n)} & \dots & c_{i,p(a \cdot n + n - 1)} \end{bmatrix} \right) \\ &= \begin{cases} \mathbf{0}, \text{ if } \mathbf{c}_{i,a} = \mathbf{0} \\ \mathbf{R}_{a-1} \oplus \cdot \beta^c, \text{ if } \mathbf{c}_{i,a} = \mathbf{1} \end{cases} \quad (7) \end{aligned}$$

For the consistent with Eq. (3), we rewrite Eq. (7) as follow:

$$\mathbf{R}_a = \mathbf{R}_{a-1} \oplus (\beta^c \& \mathbf{c}_{i,a}) \quad (8)$$

where  $\&$  stands for the bit-wise *and* operation.

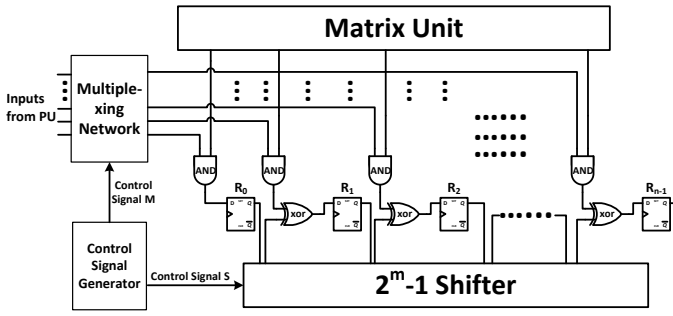


Fig. 5. Overall architecture of SR-CB-PSG

For  $0 \leq k < n$ , similar to Eq. (4), we have

$$R_k = [\hat{u}_{i-k}^c, \dots, \hat{u}_i^c] \cdot [c_{i-k,0}, \dots, c_{i,k}]^T \quad (9)$$

According to the definition of  $G$  and constituent codes, we can conduct that for any length  $n$  constituent codes, the first  $n$  columns of its corresponding rows in  $G$  should also be a generation matrix  $G_n$  for length  $n$  polar code. As described in Fig. 4b, the diagonal cycle shift is same as each correspond column, and consider the  $G_n$  is a lower triangular matrix, we get

$$\begin{aligned} & [c_{i-n+1,k+1}, \dots, c_{i-k-1,n-1}, c_{i-k,0}, \dots, c_{i,k}]^T \\ &= [c_{i-n+1,n-1-k}, \dots, c_{i,n-1-k}]^T \\ &= [0, \dots, 0, c_{i-k,n-1-k}, \dots, c_{i,n-1-k}]^T \end{aligned} \quad (10)$$

Thus, Eq. (9) can be rewritten as:

$$\begin{aligned} R_k &= [\hat{u}_{i-k+1}^c, \dots, \hat{u}_i^c] \cdot [c_{i-k,0}, \dots, c_{i,k}]^T \\ &= [\hat{u}_{i-n+1}^c, \dots, \hat{u}_i^c] \cdot [0, \dots, 0, c_{i-k,0}, \dots, c_{i,k}]^T \\ &= [\hat{u}_{i-n+1}^c, \dots, \hat{u}_i^c] \cdot [c_{i-n+1,n-1-k}, \dots, c_{i,n-1-k}]^T \\ &= \beta_{n-k-1}^c \end{aligned} \quad (11)$$

Thus, combine Eq. (11) and Eq. (8), we derive the mathematical presentation for partial sum of constituent based polar decoder as follow:

$$R_a = \begin{cases} \beta_a^c, & \text{if } a = 0 \\ R_{a-1} \oplus (\beta_a^c \& c_{i,a}), & \text{if } a \geq 1. \end{cases} \quad (12)$$

### B. Proposed architecture

According to Eq (12), the shift-register constituent-code based partial sum generator (SR-CB-PSG) is proposed as in Fig. 5. Compared with Fig. 3, there are three differences. The first difference is the input. For SR-PSG, only current estimated bit is sent into, which means the input is only from the PU from stage 0. However, for SR-CB-PSG, the inputs are from PUs of any stage, depends on the length of constituent code. Thus, a multiplexing networking is needed to route all the inputs values to the right registers. The second difference is about the shift function. According to Eq (12), instead of just shifting by one bit, the shifter should have the capability to shift  $N$ -bit where  $N$  is the length of constituent code. According to the definition of constituent code,  $N$  should be the any power of 2. Thus, A specific design  $(2^m - 1)$ -bit shifter is proposed. The control signals for both the muxing networking and shifter are from the *Control Signal Generator*(CSG) with simple logic. The last deference is matrix generation unit. For each constituent code, its corresponding  $c_{i,j}$  should be the

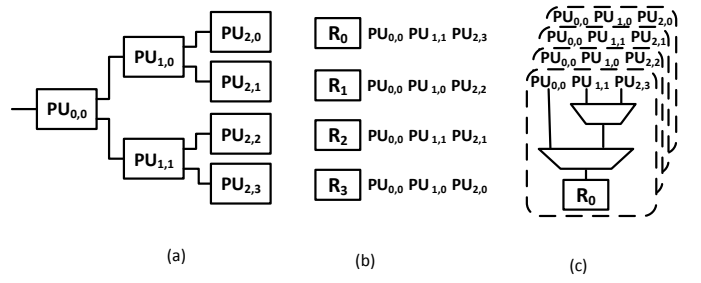


Fig. 6. (a) PU tree of SC decoder, (b) PUs and their corresponding register, and (c) architecture of multiplexing network

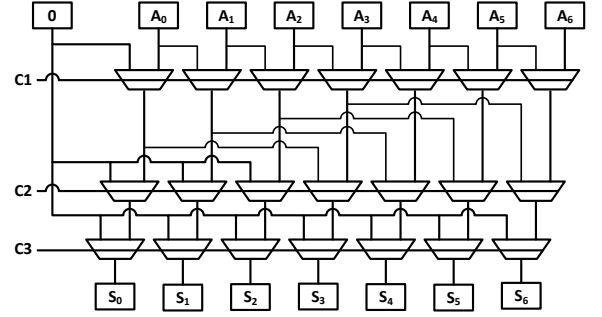


Fig. 7. An example of  $(2^m - 1)$  shifter for 16-bit polar code decoder

$i$ th row of the generation matrix, where  $i$  is the index of the last bit in the constituent code. Due to the irregularity of the constituent code, it's unnecessary to build an online generator for that. Thus, a pre-calculated ROM is placed. It's a trade-off between design complexity and hardware resource. It can be replace by a re-configurable memory device like RAM for flexibility.

Fig. 6 shows an example of partial sum routing for 8 bit constituent code based polar code. We can see each register has specific corresponding PU from each stage. The essential of multiplexing networking is to route the partial sums to the right register. For length  $N$  polar code, there are  $\log N$  stages in the decoder and  $N/2$  registers in the SR-CB-PSG. If the multiplexing networking is built from the basic 2-bit MUX, each register is assigned an identical MUXs networking made by  $(\log N - 1)$  MUXs. All the networkings share the same control signal. According to its architecture, the control signals are the direct binary mapping of its stage index. Totally,  $N/2 \cdot (\log N - 1)$  MUXs are needed. Since the multiplexer networking needs to wait each PU finish computing to get the valid inputs, it is on the critical path of the decoder. Thus, it causes additional  $\lceil \log(\log N) \rceil \cdot \Delta(MUX)$  delay, where  $\Delta(MUX)$  is the delay for a single MUX.

For  $2^{(m-1)}$  shifter, we proposed a barrel-shiftr-based architecture for that. For length  $N$  polar code,  $m \leq \log N/2$ . The shifter performs logic right shift. For  $k < n$ , where  $k$  is the index of the register and  $n$  is the length of the current constituent code, 0s are added to the left. For  $k \geq n$ , we do shift. Those behaviors satisfy the first and second in Eq (12).

Fig. 7 shows an example of  $(2^m - 1)$  shifter for 16-bit polar code decoder. All the MUXs in the same row can shall the same control signal. Those signals is generated by a  $n$  to  $2^n$  decoder, where  $n = \lceil \log(\log N) \rceil$  for length  $N$  polar

code. For length  $N$  polar code, there are  $(N/2-1) \cdot (\log N - 1)$  MUXs are needed for the shifter. Since the shifter can start shift data without waiting  $PU$  to finish computing, it is not on the critical path. Thus, it should not deteriorate the timing performance of the decoder at all.

#### IV. IMPLEMENTATION RESULTS AND DISCUSSIONS

Since no one has done PSG for constituent based SC decoder before, there is no reference design we can use. In this section, we list all the result we have and presents some relevant discussions.

Table I shows the resource consumption estimation of proposed SR-CB-PSG for length  $N$  polar code decoder and the comparison with other two conventional PSG. The most resource consumption part is the  $MUX$  since it used in both multiplexer networking and shifter. The estimation for the ROM size is based on the average calculation since the decoding latency changes along with the code rate.

TABLE I. RESOURCE COMPARISON

	proposed	[18]	[8]
DFF	$N/2$	$N$	$(N^2 - 4)/12$
MUX	$(N - 1) \cdot (\log N - 1)$	-	$N - 2$
XOR	$N/2 - 1$	$N - 2$	$N/2 - 1$
AND	$N/2$	$N/2$	-
ROM(bit)	$N^2/10(\text{average})$	-	-

The proposed design can be targeted on either ASIC or FPGA. We synthesis is both with Nangate FreePDK 45nm process and on Xilinx Kintex-7 FPGA KC705 Evaluation board. Table II shows the hardware resource of SR-CB-PSG for  $1k$  polar code decoder on both of them.

TABLE II. HARDWARE RESOURCE OF SR-CB-PSG FOR  $1k$  POLAR CODE DECODER

	XC7K325T-2FFG900C FPGA		nangate 45nm
Hardware Resource	slice LUTs	slice REGs	area
	1569(<1%)	512(<1%)	16333 $\mu m^2$

Noticeably, the architecture we discussed in this paper is based on the consideration of the worst case, which is that the maximum length of constituent codes could be  $N/2$ . However, for practical application, the maximum length of constituent is fix for certain code rate and usually cannot reach  $N/2$ . For those case, the logic of both the multiplexer networking and shifter could be even simpler, which is able to result in a better perform for both timing and area.

#### V. CONCLUSION

This paper proposes an efficient PSG for constituent code based SC decoder. Conventional PSG is not compatible with the constituent code based SC decoder. This is because that the conventional one is only capable of taking estimated bit one by one however the constituent code based decoder is generating the intermediate partial sum directly. Aim to solve this problem, we first derive the mathematical presentation for constituent code based PSG from the SR-PSG for conventional SC decoder. Then, the overall hardware architecture and design details are proposed. Finally, the implementation result with both VLSI and FPGA technology are presented, and the relevant discussions are also mentioned as well.

#### REFERENCES

- [1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *Information Theory, IEEE Transactions on*, vol. 55, no. 7, pp. 3051–3073, 2009.
- [2] I. Tal and A. Vardy, "List decoding of polar codes," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 1–5.
- [3] N. Hussami, S. B. Korada, and R. Urbanke, "Performance of polar codes for channel and source coding," in *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1488–1492.
- [4] J. Xu, T. Che, and G. Choi, "Xj-bp: Express journey belief propagation decoding for polar codes," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.
- [5] B. Yuan and K. K. Parhi, "Architecture optimizations for bp polar decoders," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2654–2658.
- [6] J. Lin, J. Sha, L. Li, C. Xiong, Z. Yan, and Z. Wang, "A high throughput belief propagation decoder architecture for polar codes," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 153–156.
- [7] C. Leroux, A. J. Raymond, G. Sarkis, and W. J. Gross, "A semi-parallel successive-cancellation decoder for polar codes," *Signal Processing, IEEE Transactions on*, vol. 61, no. 2, pp. 289–299, 2013.
- [8] C. Zhang and K. Parhi, "Low-latency sequential and overlapped architectures for successive cancellation polar decoder," *Signal Processing, IEEE Transactions on*, vol. 61, no. 10, pp. 2429–2441, 2013.
- [9] B. Yuan and K. K. Parhi, "Low-latency successive-cancellation polar decoder architectures using 2-bit decoding," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1241–1254, April 2014.
- [10] —, "Low-latency successive-cancellation list decoders for polar codes with multibit decision," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2268–2280, Oct 2015.
- [11] T. Che, J. Xu, and G. Choi, "Overlapped list successive cancellation approach for hardware efficient polar code decoder," *arXiv preprint arXiv:1511.00577*, 2015.
- [12] A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 61, no. 8, pp. 609–613, 2014.
- [13] A. Balatsoukas-Stimming, M. Bastani Parizi, and A. Burg, "Llr-based successive cancellation list decoding of polar codes," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. Ieee, 2014, pp. 3903–3907.
- [14] A. Alamdar-Yazdi and F. R. Kschischang, "A simplified successive-cancellation decoder for polar codes," *IEEE communications letters*, vol. 15, no. 12, pp. 1378–1380, 2011.
- [15] G. Sarkis, P. Giard, A. Vardy, C. Thibeault, and W. J. Gross, "Fast polar decoders: Algorithm and implementation," *Selected Areas in Communications, IEEE Journal on*, vol. 32, no. 5, pp. 946–957, 2014.
- [16] T. Che, J. Xu, and G. Choi, "Tc: Throughput centric successive cancellation decoder hardware implementation for polar codes," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 991–995.
- [17] J. Lin and Z. Yan, "A hybrid partial sum computation unit architecture for list decoders of polar codes," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1076–1080.
- [18] G. Berhault, C. Leroux, C. Jogo, and D. Dallet, "Partial sums generation architecture for successive cancellation decoding of polar codes," in *SiPS 2013 Proceedings*. IEEE, 2013, pp. 407–412.
- [19] —, "Partial sums computation in polar codes decoding," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 826–829.
- [20] Y. Fan and C.-y. Tsui, "An efficient partial-sum network architecture for semi-parallel polar codes decoder implementation," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3165–3179, 2014.